Identifying SNPs without a reference genome by comparing raw reads

Pierre Peterlongo, Nicolas Schnel, Nadia Pisanti, Marie-France Sagot, Vincent Lacroix

Identifying SNPs without a reference genome by comparing raw reads

Pierre Peterlongo[1], Nicolas Schnel[1], Nadia Pisanti[2], Marie-France Sagot[3], and Vincent Lacroix[3]

[1] INRIA Rennes - Bretagne Atlantique, EPI Symbiose, Rennes, France
[2] Dipartimento di Informaticà, Universita di Pisa, Italy
[3] INRIA Rhône-Alpes, 38330 Montbonnot Saint-Martin, France and Université de Lyon, F-69000 Lyon; Université Lyon 1; CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, F-69622 Villeurbanne, France

Abstract. Next generation sequencing (NGS) technologies are being applied to many fields of biology, notably to survey the polymorphism across individuals of a species. However, while single nucleotide polymorphisms (SNPs) are almost routinely identified in model organisms, the detection of SNPs in non model species remains very challenging due to the fact that almost all methods rely on the use of a reference genome. We address here the problem of identifying SNPs without a reference genome. For this, we propose an approach which compares two sets of raw reads. We show that a SNP corresponds to a recognisable pattern in the de Bruijn graph built from the reads, and we propose algorithms to identify these patterns, that we call *mouths*. We outline the potential of our method on real data. The method is tailored to short reads (typically Illumina), and works well even when the coverage is low where it reports few but highly confident SNPs. Our program, called kisSnp, can be downloaded here: http://alcovna.genouest.org/kissnp/.

1 Introduction

Biology in general, and genomics more particularly, witnessed a revolution in the middle 1970s with the development of rapid DNA sequencing techniques, notably the Sanger method which remained the standard approach for sequencing including whole genomes until the early years of the twenty first century. We have since then been witnessing a second revolution, various orders of magnitude bigger than the first, with the advent of the so-called "next generation sequencers" (NGS for short) which enable to obtain up to several hundred million bases in one single run at increasingly lower costs. These include (not exclusively) the 454 Life Sciences, SOLiD Applied Biosystems and Illumina technologies, each with its own characteristics in terms of read length and error rate. Such characteristics are however evolving extremely fast, faster indeed than the algorithms developed to handle the data such technologies produce.

This incredible acceleration has two implications that motivate the work presented in this paper: first it is now possible to obtain data for various individuals of a same species and thus to investigate the genetic differences among such individuals, and second, increasingly more often this will concern species for which

we have no genome of reference, that is no genome already fully sequenced and assembled that could guide the investigation.

The genetic markers that will be of interest in this paper are so-called Single Nucleotide Polymorphisms (SNP for short). These correspond to a DNA sequence variation that occurs when a single nucleotide – A, T, C, or G – in a genome differs among members of a species or between paired chromosomes in an individual. There are two types of SNPs: substitutions or insertions/deletions. We focus here on the first type, that is on substitutions of single nucleotides.

Identifying SNPs in a population may have a wide range of applications that goes from assessing the polymorphism of the population, linking this polymorphism to phenotype information, or selecting SNPs as markers of subpopulations. However, while SNPs are almost routinely identified in model organisms, the detection of SNPs in non model species remains very challenging due to the fact that almost all methods to identify SNPs rely on the use of a reference genome.

Our objective is, given high-throughput read data for a pair of individuals, to identify a set of SNPs with good confidence, without having to perform an assembly of the reads with all the possible mistakes this entails, in a context where we do not have a reference sequence to help the identification.

We are aware of only two publications, dating both from 2010, that deal with the same problem [3, 9]. Recognisably, the major difficulty one faces is due to the presence of errors in the reads, which may be mistaken for a SNP. Additionally, the presence of inexact repeats in the genomes of the studied individuals, may further harden the task. In this paper, we restrict to the case where there is only one genomic variant for each individual (we say that the individuals are homozygous). In this context, the issue of repeats is greatly reduced.

Ratan *et al.* [9] first filter the reads in order to remove the repetitive sequences, then create clusters of overlapping reads which they assemble using a short read assembler. The SNPs are finally identified in the micro-assembled regions using a combination of filters, based on the number of reads supporting each variant or the distance of the SNP w.r.t. the end of the contig.

Unlike Ratan *et al.*, we chose not to use an assembler, which we think can make undesired choices as to sequence variants to remove during the assembly. Indeed, the purpose of an assembler is not to identify SNPs but to propose one reference sequence compatible with the data. Similar to Canon *et al.* [3], we work with raw reads, but we go further than a statistical description of the reads and propose to locally reconstruct the de Bruijn graph in order to identify SNPs. The use of a de Bruijn graph in computational biology was introduced by Pevzner *et al.* in 2001 [8] and used since then as a first step by many short read assemblers.

The key point of our method is that a SNP corresponds to a recognisable pattern in the de Bruijn graph, which we call a *mouth*, each *lip* of the mouth representing an individual variant of a same genomic locus.

Our aim is to directly find the mouths that may be reliably associated to a SNP without making use of any preliminary filters that may eliminate repeats. This is important because, although not explicitely stated, such filters seem to strongly rely on the assumption of an approximately uniform coverage of

each sequence position by the reads in the available data. This assumption is usually not true. Moreover, the biases in read coverage may even vary across two sequencing experiments of the same genomic sample. This means that filters may remove sequences which in fact do not belong to repeats.

We thus present in this paper an algorithm which takes as input two sets of short reads (Illumina or AB/SOLiD) and outputs candidate SNPs (i.e. mouths in the de Bruijn graph), without performing any filtering nor using a short read assembler. This is what we call a comparative micro-assembly. This method is new as, as far as we know, no other treats data coming from distinct sequencing experiments. This approach presents the interest of taking advantage of differences in the data directly into the heart of the algorithm and not in a post-treatment step. SNPs are thus detected on raw read data instead of on pre-assembled sequences. We applied our algorithm on data simulated using MetaSim [10], where we show under which sets of parameters the method works best. We finally apply the method to real data for *Escherichia coli*, for which experimentally validated SNPs are available [2], which is very rare. We show that our method successfully identifies the previously known SNPs, but also predicts new SNPs missed by the conservative method used in the original publication [2].

2 Preliminaries

Sequence, k-mers, prefix, suffix. A *sequence* is composed by zero or more symbols from an alphabet Σ containing $|\Sigma|$ distinct characters. A sequence s of length n on Σ is denoted also by $s[0]s[1]\dots s[n-1]$, where $s[i] \in \Sigma$ for $0 \leq i < n$. The length of s is denoted by $|s|$. Finally, we denote by $s[i,j]$ the *substring* $s[i]s[i+1]\dots s[j]$ of s. In this case, we say that the substring $s[i,j]$ occurs at position i in s. We call *k-mer* a substring of length k. If $s = uv$ for $u, v \in \Sigma^*$, we say that v is a *suffix* of s and that u is a *prefix* of s.

De Bruijn graph. Each node of a de Bruijn graph stores exactly one k-mer. An edge connects a node n_0 to a node n_1 if the suffix of length $k-1$ of the k-mer corresponding to node n_0 is equal to the prefix of length $k-1$ of the k-mer corresponding to node n_1.

A category of *de novo* read-assembly methods such as SOAPdenovo [7], Euler [8] and Velvet [11] (to mention a few) uses the de Bruijn graph as a fundamental data structure. In a few words, reads are first divided into overlapping k-mers, then the associated de Bruijn graph is created and finally Eulerian paths are found in the graph for reconstructing the initial genomic sequence, or fragments thereof that are as large as possible (contigs).

One of the main difficulties encountered by such methods comes from the sequencing errors that generate substitutions and insertions/deletions in the data that must then be assembled. Such errors lead to loops in the de Bruijn graph which may hinder the Eulerian path detection. A first step in such algorithms consists thus in "cleaning the data" by removing suspicious reads and substituting suspect nucleotides. This cleaning step may be problematic when looking

for SNPs as it may remove a significant part of them that will be mistakenly considered as sequencing errors.

3 Comparative micro-assembly model

Our method compares reads generated by two distinct sequencing experiments, and creates parts of the de Bruijn graph potentially linked to a SNP between these two experiments, thereby detecting such SNPs.

The main idea is that the de Bruijn graph of k-mers stemming from two sequences that contain a SNP presents a mouth shape as shown in Fig. 1. The algorithm described in Section 4 detects and constructs such graph shapes directly from the non-assembled k-mers coming from the sets of reads of two distinct sequencing experiments. It is important to notice that the algorithm does not reconstruct the full de Bruijn graph but focuses only on putative SNPs by building mouths.

Mouth model definition. In a de Bruijn graph of k-mers coming from reads of two sequencing experiments (reads A and reads B), a *mouth* is composed by:

- an upper path of k overlapping k-mers $\{a_0..a_{k-1}\}$ resulting from the reads of at least set A. This path is called the *upper lip* of the mouth;
- a lower path of k overlapping k-mers noted $\{b_0..b_{k-1}\}$ resulting from the reads of at least set B. The a_i's and the b_i's differ by one substitution. This path is called the *lower lip* of the mouth;
- a left (resp. right) node, noted c_{-1} (resp. c_k) that corresponds to a k-mer present in both sets A and B and is connected to both a_0 and b_0 (resp. a_{k-1} and b_{k-1}). These k-mers are called the *closing k-mers* of the mouth.

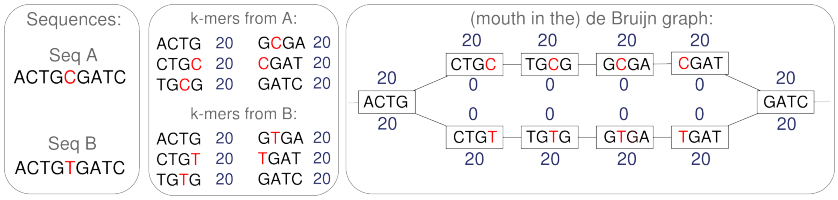

Fig. 1. A SNP between two genome fragments (Seq A and Seq B) generates a mouth shape (rightmost frame) in the de Bruijn graph of the k-mers (here $k = 4$) extracted from Seq A and Seq B. It is assumed in this example that the coverage is exactly 20 (each position of each sequence is covered by 20 reads, thus each position gives rise to 20 k-mers). In the rightmost frame, the number above (resp. below) the nodes indicates the number of occurrences in Seq A (resp. Seq B).

Taking into account the *k*-mer counts. Let us first consider the case where the sequencing is perfect: uniform coverage $\mathcal{C}$ and no sequencing errors nor repeats. In such case, all k-mers from A covering a SNP have $\mathcal{C}$ occurrences more than the same k-mers from B, and *vice-versa*. We considered this theoretical perfect coverage $\mathcal{C} = 20$ in the example of Fig. 1. In practice, the coverage is not uniform and the reads contain errors. The consequence is that the k-mer count difference between experiments will not be constant along the mouth. To account for this, we introduce a parameter called δ, which is meant to capture a deviation from the exact case. Below, we describe the mouth model with counts.

Mouth model integrating k-mer counts. For any k-mer ω, let $mult_A(\omega)$ (resp. $mult_B(\omega)$) be its number of occurrences in the set of reads A (resp. B). We define $diff(\omega) = mult_A(\omega) - mult_B(\omega)$. The SNP mouth model integrates the k-mer number of occurrences as follows. A special k-mer a_{op} called the *opening k-mer* is chosen as reference (see Section 4). If a_{op} is in the upper (resp. lower) lip, for any k-mer a_i contained in a node of the upper (resp. lower) lip, we have $diff(a_i) = diff(a_{op}) \pm \delta$ while for any k-mer b_i contained in a node of the lower (resp. upper) lip, we have $diff(b_i) = -diff(a_{op}) \pm \delta$. The left and right closing k-mers c_{-1} and c_k have no counting properties.

4 Algorithm kisSnp for mouth detection

Algorithm outline. The algorithm kisSnp takes as input two sets of reads (A and B) coming from two distinct sequencing experiments. The output is a set of pairs of micro-assembled sequences, each of length $2k-1$, differing by exactly one substitution located at the central position. Those correspond to putative SNPs detected thanks to the mouth model. The algorithm is divided into three main steps:

- For each set A and B, extract the k-mers and their reverse complement and store them in a tree together with their number of occurrences.
- Create the mouths (detailed in Section 4.1):
  - For each possible opening k-mer a_{op}, detect all possible opposite opening k-mers b_{op} distant by one substitution from a_{op} and fulfilling the counting model.
  - For each pair (a_{op}, b_{op}), construct the mouth by extending the k-mers to the right and to the left with coherent k-mers (*i.e.* overlapping on $k-1$ characters and fulfilling the counting model).
  - Stop the right and left extensions once the mouth is closed or no extension can be found.
- Check that the found mouths are coherent with the reads (detailed in Section 4.3).

4.1 Creating the mouths

Selection of the k-mers opening the mouth. The opening k-mer a_{op} is selected such that $max\,(mult_A(a_{op}), mult_B(a_{op})) < max\,(mult_A(a_i), mult_B(a_i))$ for any k-mer $a_i \neq a_{op}$ and $diff(a_{op}) \neq 0$. In other words, a_{op} is the k-mer having the smallest number of occurrences in either set A or B (possibly zero occurrence in one of the two sets), and is such that it occurs more in a set than in the other, possibly due to a SNP. The rationale for choosing the k-mer with the smallest count is to avoid choosing a k-mer involved in a repeat for opening the mouth. The opposite opening k-mer b_{op} is selected such that a_{op} and b_{op} are distant by exactly one substitution and $diff(b_{op}) = -diff(a_{op}) \pm \delta$. Notice that the substitution position between the k-mers a_{op} and b_{op} may be anywhere. We denote by p this substitution position ($p \in [0, k-1]$). It is worth noticing that, for a given opening k-mer a_{op}, several (at most $(|\Sigma| - 1)k$) distinct k-mers b_{op} may satisfy these conditions. They are all iteratively tested as mouth openers.

Extending and closing the mouth. Once a pair of opening k-mers (a_{op}, b_{op}) is selected, a recursive procedure extends them to the right and left with other k-mers fulfilling the following conditions (also shown in Fig. 2). The k-mers a_{i+1} and b_{i+1} may extend a_i and b_i iff:

- $p \geq 0$ (the closing k-mer c_k has not yet been reached), and
- $a_i[1, k-1] = a_{i+1}[0, k-2]$ and $b_i[1, k-1] = b_{i+1}[0, k-2]$ (the new k-mers overlap on $k-1$ characters with their predecessors), and
- $a_{i+1}[k-1] = b_{i+1}[k-1]$ (the extension is done with a same character), and
- $diff(a_{i+1}) = diff(a_{op}) \pm \delta$ and $diff(b_{i+1}) = -diff(a_{op}) \pm \delta$ (the counting model is fulfilled).

Similar conditions apply to a_{i-1} and b_{i-1} for extending a_i and b_i on the left.

The two lips of a mouth have to be closed. A mouth can be right-closed (resp. left-closed) if there exists a k-mer c_k (resp. $c_{-1)}$) whose prefix (resp. suffix) of length $k-1$ is equal to the suffix (resp. prefix) of length $k-1$ of a_{k-1} (resp. a_0), by definition itself equal to the suffix (resp. prefix) of length $k-1$ of b_{k-1} (resp. b_0). Once the mouth is right- and left- closed, the procedure stops.

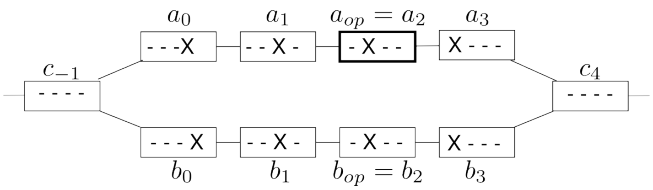

Fig. 2. A mouth with $k = 4$. The symbol $'-'$ stands for a match between two k-mers positions while the symbol $'X'$ stands for a mismatch. The k-mer a_2 is the opening k-mer and the k-mer b_2 is the opposite opening k-mer. Here, $p = 1$ as $a_{op}[1] \neq b_{op}[1]$.

Disabling the use of a same opening k-mer more than once Once a k-mer a_{op} was used for opening a mouth, it is flagged and never used anymore either as an opening or as an extending k-mer. The underlying idea is to avoid detecting twice the same mouth. Moreover, we can safely discard this k-mer because, since it was the one with smallest count, it should not belong to another mouth.

4.2 Complexity

The time complexity can then be divided into two main parts as follows.

- **Indexation:** if N is the total number of distinct k-mers, indexing them into a tree can be done in $O(N \log N)$ time using heap sort. This then provides access in time $O(k)$ to any k-mer information.
- **Mouth creation:** Given one opening k-mer a_{op}, at worst $k \times (|\Sigma| - 1)$ k-mers b_{op} may fulfill the opening conditions. Any k-mer may be opening. Thus at worst, $O(N \times k \times |\Sigma|)$ mouth opening pairs must be tried. Given an opening pair of k-mers a_{op} and b_{op}, in the worst case, for each of the $k-1$ steps of the extension, $|\Sigma|$ k-mers muts be tested. Access to a k-mer information being in time $O(k)$, the time complexity for one mouth extension is thus $O(k|\Sigma|^k)$. Thus, at worst, the time complexity for mouth finding is $O\left(N \times k^2 \times |\Sigma|^{2k}\right)$.

Each k-mer being stored in $O(k)$, the space complexity is $O(Nk)$.

4.3 Checking for read coherence

Two k-mers are linked in the de Bruijn graph if they overlap over k-1 characters, without checking whether the created k+1-mer indeed exists in the set of reads. This may lead to false-positive results. In order to remove those k-mers, in a post-treatment step, we check for *read-coherency* of the identified mouths. The upper (resp. lower) lip of a mouth is said to be read-coherent if it is 100% covered by reads from set A (resp. B) and, moreover, if in the upper (resp. lower) lip the SNP position is covered by at least two distinct reads from set A (resp. B). We keep only the mouths for which both lips are read-coherent. The rationale for restricting to mouths covered by at least 2 reads is to minimize the number of sequencing errors that are mistaken for SNPs. Indeed, it is unlikely that a sequencing error occurs at the same nucleotide for 2 distinct reads, as shown in [9].

5 Applications to simulated read data

We developed the algorithm in a program called kisSnp, coded in Java, that was used for testing our approach. kisSnp is available for download at: http://alcovna.genouest.org/kissnp/.

To test our approach on controlled datasets, we applied the following procedure. Given an input sequence s_{ref}, we generated a sequence s_{snp} such that

s_{ref} and s_{snp} differ by $\left\lfloor \frac{|s_{ref}|}{1000} \right\rfloor$ substitutions, each considered as a SNP. The average distance between two virtual SNPs is then 1000 nucleotides, in agreement with [4]. The substitutions are randomly distributed over s_{snp}, and each substitution is introduced following the transition/transversion probabilistic model [5].

The sequence s_{ref} and s_{snp} are then virtually sequenced into a set of reads r_{ref} and r_{snp} using the MetaSim [10] program. Among other parameters, MetaSim enables to tune the sequencing errors model as well as the average coverage. In all our experiments, we generated reads of length 62, in agreement with the Illumina technology. kisSnp is then tested using sets r_{ref} and r_{snp}.

5.1 Finding the SNPs

Human chrX portion We extensively tested the parameters of kisSnp on a small portion of the human chromosome X of length 137897 *bp*, corresponding to a kinesin family member 4A (KIF4A). We applied MetaSim to the pair s_{ref}, s_{snp} distant by 137 simulated SNPs with i) no sequencing errors (Fig. 3(a)) and ii) errors following an empirical Illumina error model (Jean Marc Aury, Genoscope, personal communication – Fig. 3(b) and (c)).

One may start by observing that the quality of the results is relatively robust to the choice of parameters. With a good coverage ($> 4x$), large distinct sets of parameters thus enable to find almost all SNPs with no false positives. The main lessons learnt about such parameters from these results are the following:

- The δ value has not a strong influence for $\delta \geq 20$ (Fig. 3(b) vs Fig. 3(c)). However, for smaller values of δ (data not shown), the specificity decreases rapidly due to sequencing errors and a non uniform coverage.
- As expected, for small values of k (≤ 20), false positives are found. For larger values of k (say, ≥ 30), less SNPs are found as more k-mers involve sequencing errors and/or more positions are not covered by any k-mer.

Concerning the data, coverage is of major importance as a small coverage leads to lower sensitivity (in each case, the lower the coverage, the less sensitive is kisSnp). Illumina sequencing errors have a small influence on the results (Fig. 3(a) vs Fig. 3(b)). One important message is that, using $k = 25$ and $\delta \geq 20$, all experiments obtained 100% specificity (no false positives) for various values of sensitivity, the latter depending in particular on the coverage.

Neisseria meningitidis strain MC58 One main limitation of our mouth model could be the presence of repeats in the genomes considered. To measure the effect of repeats on the performance of kisSnp, we performed experiments on the bacterium *Neisseria meningitidis* (strain $MC58$) of length 2.27 Mbp. The size of the repeated elements in this genome range from 10 bases to more than 2000 bases, and their number may reach more than 200 copies. We performed tests on the original MC58 sequence introducing 2272 SNPs and simulating reads using MetaSim with an Illumina error model. Moreover, we performed exactly the same tests on a randomised sequence obtained from the MC58 sequence

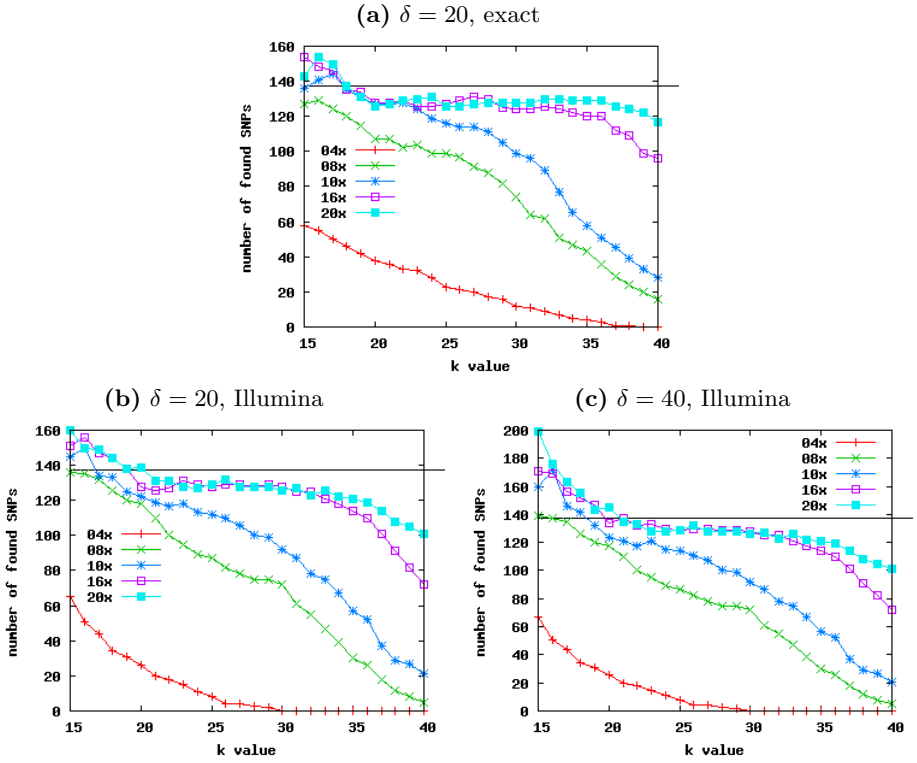

Fig. 3. Number of SNPs (read-coherent mouths) found by kisSnp. Fragment of the human chromosome X, 137 SNPs to find (symbolised by the horizontal line). The "$n\times$" values indicate the coverage used while simulating the reads.

(same length and nucleotide frequencies, distribution of nucleotides following a Bernoulli model). Results for both MC58 and the randomised MC58 are presented in Fig. 4.

One may observe that even on a difficult dataset like MC58, a large part of the SNPs are identified (26%, 86% and 97% respectively with coverage 4x, 10x, and 20x with $k = 20$). Another important remark is that the difference between the results obtained on MC58 and the randomised MC58 is small, showing that the algorithm is robust to repeats.

5.2 Execution time

The kisSnp program, coded in Java, is a prototype and is not yet time optimised. The performance results below enable only to give an idea of the evolution of the running time with different parameters. All tests were done on a DELL laptop, quad-core 2.67GHz with 4Gb memory running under Fedora Core 12.

We started by testing on the human KIF4A dataset (simulating reads with an Illumina error model), the influence of the δ parameter on the execution time. We observed (data not shown) that this has no influence whatsoever for $\delta \geq 20$.

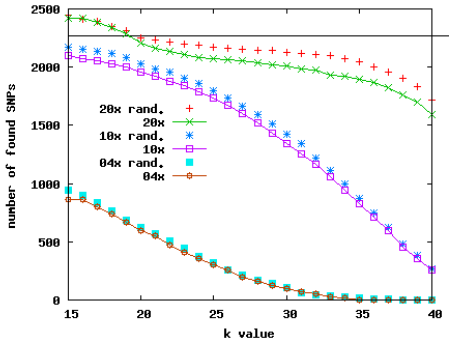

Fig. 4. Results on MC58 and the randomised MC58 sets while looking for 2272 SNPs (horizontal line) with $\delta = 20$. The "$n\times$" values indicate the coverage used while simulating the reads, "*rand.*" stands for results on the randomised MC58 set.

On the same dataset, we fixed $\delta = 30$ and checked the execution time for values of k varying from 0 to 40. The results, presented in Fig. 5, show that two phases may be distinguished. For k from 0 to 6, we observe an exponential time growth that is in agreement with the theoretical complexity. During this phase, the worst-case behaviour is reached, each mouth extension tests $|\Sigma|^k$ lips. The second phase is observed for bigger values of k, when a large number of possible k-mers are no longer present in the data, thus limiting the number of tested lips extensions. This greatly reduces the execution time, which starts decreasing for $k \geq 25$ as less and less mouths are successfully created.

The execution time also highly depends on the number N of distinct k-mers we have to deal with. We thus performed experiments on random sequences of growing size. The results are presented in Fig. 6. The execution time grows linearly with N while N remains below a threshold of around 15 million reads. Above this threshold, one observes an exponential growth that could be explained by the fact that the kisSnp prototype uses a hash table instead of a tree for storing and accessing the k-mers information. With a large number of k-mers, the hash table load factor becomes higher than 0.75 increasing the lookup cost, because of an important number of collisions.

6 Applications to real read data

To test our approach on a real dataset now, we used raw reads from the *Escherichia coli Long-term Experimental Evolution Project* [1] whose purpose was to grow *Escherichia coli* during more than 20 years, conserving a sample each 500 generations. The SNPs found over these generations are listed in the Barrick *et. al* paper [2]. An Illumina 1G platform was used for sequencing the samples with reads of length 36 and a high coverage ($50x$). We focused our attention on the raw reads from the first generation sample, and those from the

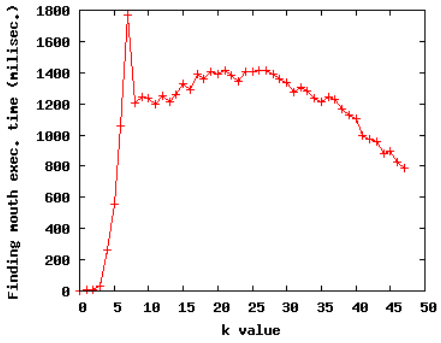

Fig. 5. Influence of k on the execution time. Data: set KIF4A (described Section 5.1), $\delta = 30$.

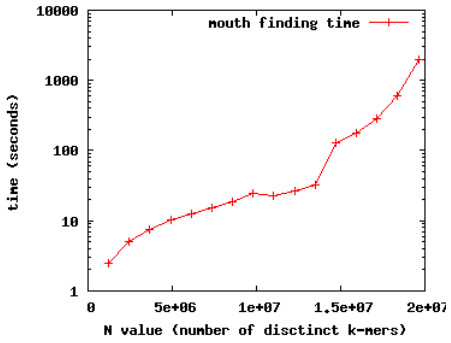

Fig. 6. Execution time with respect to the number N of distinct k-mers of length $k = 25$, with $\delta = 20$, Bernoulli random sequences.

20.000^{th} generation sample. The existence of experimentally validated SNPs is very rare which is the main reason that led us to work on this dataset, for which true positives are known.

Using a custom-made computational pipeline called BRESEQ, Barrick *et. al* identified 28 SNPs by mapping these two generations of reads against the reference genome CP000819. These 28 SNPs were then experimentally validated.

We used kisSnp for comparing these two sets of reads, forgetting the reference genome. Using as parameters $k = 26$ and $\delta = 20$, 88 SNPs were found. Of these, 27 of the 28 SNPs found by Barrick *et. al* were also identified by us, giving a sensitivity of 96%. Our kisSnp method missed one SNP, located position 430835.

To evaluate the potential interest of the remaining 61 SNPs we identified, we mapped them against the reference genome using Maq [6]. Among the 61, 43 correspond to a SNP structure not detected in the Barrick *et. al* project: the two lips map at the same position with one substitution in the 20.000^{th} generation sequence. The remaining 18 correspond to suspicious SNPs. Indeed, the two lips do not map to the same position in the genome. Without experimental validation, one however cannot conclude on those 61 detected putative SNPs.

The results obtained with kisSnp are very good on this real dataset, as without a reference genome it was able to find back 27 of the 28 experimentally verified SNPs, and 41 additional ones that correspond to real SNP structures.

7 Conclusion and future work

We proposed an algorithm for comparing the raw outputs of short reads experiments, typically Illumina ones, with the purpose of finding SNPs between individuals of a same species. This is of particular interest for quickly designing genomic tags without waiting and/or paying for a full genome assembly.

Preliminary results on both simulated and real experimental data are particularly promising. In both cases, kisSnp identifies the SNPs, while not being too

sensitive to the parameters used. On a real dataset, kisSnp enabled to find 96% of the SNPs initially detected by mapping to the reference genome. In addition, we propose new SNPs, which could be tested experimentally.

There is clearly room for improvement. For now, the method does not handle heterozygous SNPs, does not take sequencing qualities into account and is limited to pairwise comparison while sets of more than two individuals may be compared. More generally, the three challenges of SNP identification are that the reads contain errors, the genome contains repeats and the read coverage is not uniform. This last item is usually disregarded whereas we notice that it has a significant impact on the results. We expect that several algorithms in the area of NGS bioinformatics could be improved by taking this observation into account.

Acknowledgments

We are grateful to Jean Marc Aury, who provided the empirical Illumina error model, to Matteo Brilli who pointed out to us the *E.coli* experiment, to the GenOuest platform who supported extensive computations and to the INRIA ARC Alcovna for financial support.

References

1. E. coli long-term experimental evolution project site, http://myxo.css.msu.edu/ecoli/.
2. J. E. Barrick, D. S. Yu, H. Jeong, T. K. Oh, D. Schneider, R. E. Lenski, and J. F. Kim. Genome evolution and adaptation in a long-term experiment with escherichia coli. *Nature*, 461:1243–1247, 2009.
3. C. Cannon, C.-S. Kua, D. Zhang, and J. Harting. Assembly free comparative genomics of short-read sequence data discovers the needles in the haystack. *Molecular Ecology*, 19(Suppl. 1):147–161, 2010.
4. D. N. Cooper, B. A. Smith, H. J. Cooke, S. Niemann, and J. Schmidtke. An estimate of unique DNA sequence heterozygosity in the human genome. *Hum. Genet.*, 69:201–205, 1985.
5. M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120, Dec 1980.
6. H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, 2008.
7. R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, 20(2):265–272, 2010.
8. P. Pevzner, H. Tang, and M. Waterman. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.*, 98:9748–9753, 2001.
9. A. Ratan, Y. Zhang, V. Hayes, S. Schuster, and W. Miller. Calling SNPs without a reference genome. *BMC Bioinformatics*, 11:130, 2010.
10. D. Richter, F. Ott, A. Auch, R. Schmid, and D. Huson. MetaSim – A Sequencing Simulator for Genomics and Metagenomics. *PLoS ONE*, 3(10):e3373, 2008.
11. D. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, 18(5):821–829, 2008.